Relatório TP SO

Trabalho Prático de SO

Engenharia de Sistemas Informáticos

Ano Letivo 2021/22

Realizador por:

Rúben Gomes nº21118

João Furtado nº21125

Eduardo Rebelo nº21105

Rui Miranda nº21135

André Ferreira nº21147

João Castro nº21143

Conteúdo

Relatório TP SO 1
Contribuição de cada elemento. Parte 1 3
Contribuição de cada elemento. Parte 2 3
Parte 1 4
Introdução 5
Resolução 6
Alínea a) 6
Alínea b) 7
Alínea c) 8
Alínea d) 9
Alínea e) 10
Alínea f) 11
Alínea g) 12
Menu 13
Parte 2 14
Introdução 15
Resolução 16
Alínea a) 16
Alínea b) 17
Alínea c) 18
Alínea d) 19
Alínea e) 20
Alínea f) 20
Conclusão 21

Contribuição de cada elemento. Parte 1

Rúben Gomes nº21118- f), g), d)

João Furtado nº21125- c), e), f)

Eduardo Rebelo nº21105- a), b), g)

Rui Miranda nº21135- d), a)

nº21147- e)

nº21143- c)

Contribuição de cada elemento. Parte 2

Rúben Gomes nº21118- a), b), c), d), e), f)

João Furtado nº21125- a), b), c), d), e), f)

Eduardo Rebelo nº21105- a), b), c), d), e), f)

Rui Miranda nº21135- a), b), c), d), e)

André Ferreira nº21147- a), b), c), d), e)

João Castro nº21143- a), b), c), d), e)

Parte 1

Introdução

Para a Parte 1 do trabalho prático foi-nos proposto desenvolver um conjunto de comandos com funções de chamada ao sistema (system calls). O seguinte relatório tem como objetivo explicar e exemplificar como foi realizado o desenvolvimento deste conjunto de comandos, e apresentar as nossas dificuldades.

Resolução

Alínea a)

Função **mostra** ficheiro: Nesta função o objetivo era imprimir na consola o ficheiro pedido pelo utilizador. Para esta função simplesmente abrimos o ficheiro com a função open(), escrevendo o seu conteúdo no array content, que depois imprimimos na consola.

```
22  /// Função utilizada para mostrar no ecrã todo o conteúdo de um ficheiro
23  int mostraFicheiro(char fileName[50])
24  {
25      int fd, len;
26      char content[MAX];
27      fd = open(fileName, O_RDONLY);
28      if (fd == -1)
29      {
30          write(STDERR_FILENO, "Ficheiro não existe", 20);
31      }
32      len = read(fd, content, sizeof(content));
33      write(STDOUT_FILENO, content, len);
34      close(fd);
35
36      return 1;
37  }
38
```

Alínea b)

Função **copia** ficheiro: Para copiar um ficheiro para o outro, pedimos ao utilizador o ficheiro que desejava copiar. Com essa informação, abrimos o ficheiro pedido, e criámos um ficheiro onde imprimimos o conteúdo do ficheiro original.

```
104  /// Função utilizada para criar um ficheiro com o conteúdo de outro
105  void copiarFicheiro(char fileName[50])
106  {
107
108      char copia[100];
109
110      int ficheiroI, ficheiroC, leitura;
111
112
113      ficheiroC = creat("ficheiro.copia.txt", S_IRWXU);
114
115      ficheiroI = open(fileName, O_RDONLY);
116
117      if (ficheiroI == -1)
118      {
119          write(STDERR_FILENO, "Ficheiro não existe", 20);
120      }
121
122      leitura = read(ficheiroI, copia, sizeof(copia));
123
124      write(ficheiroC, copia, leitura);
125
126      close(ficheiroI);
127      close(ficheiroC);
128  }
129
```

Alínea c)

Função **acrescenta** ficheiro: Aqui foi pedido como parâmetro dois nomes de ficheiros, os quais temos de juntar. Abriram-se os 2 ficheiros, o que tem de ser copiado só com permissões de leitura. Já o segundo fez uso do "O_APPEND", que indicava o apontador para o fim do ficheiro.

A partir daí bastou imprimir cada caracter de um ficheiro para o outro usando o ciclo *while* apresentado.

```
39  /// Função utilizada para acrescentar num ficheiro o conteúdo de outro
40  void acrescentaFicheiro(char fileName1[50], char fileName2[50])
41  {
42      int fd1, fd2;
43      char c;
44      fd1 = open(fileName1, O_RDONLY);
45      fd2 = open(fileName2, O_APPEND | O_WRONLY);
46
47      if (fd1 == -1 || fd2 == -1)
48      {
49          write(STDERR_FILENO, "Ficheiro não existe", 20);
50      }
51
52      while (read(fd1, &c, sizeof(c)))
53      {
54          write(fd2, &c, sizeof(c));
55      }
56
57      close(fd1);
58      close(fd2);
59  }
60
```

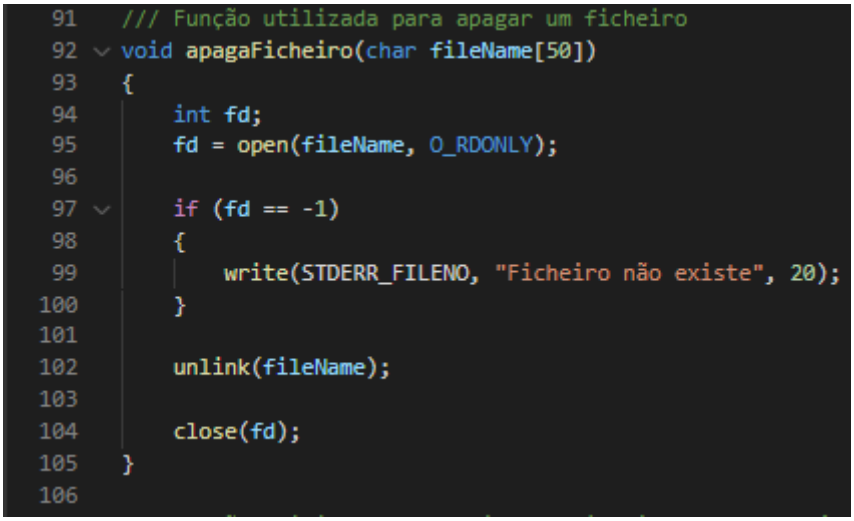
Alínea d)

Função **conta** ficheiro: Para esta função, criámos um ciclo que percorre todo o ficheiro com a ajuda da variável “len”. Desta maneira, é possível percorrer cada caracter do ficheiro e verificar se algum deles é “\n”. Se for um “\n”, isto quer dizer que encontramos uma nova linha, aí incrementamos a variável que conta as linhas e retornamo-la no fim da leitura.

```
61  /// Função utilizada para contar as linhas de um ficheiro
62  int contaLinhas(char fileName[50])
63  {
64      char buffer[MAX];
65      int len;
66      unsigned lines = 1;
67
68      int fd1 = open(fileName, O_RDONLY);
69
70      if (fd1 == -1)
71      {
72          write(STDERR_FILENO, "Ficheiro não existe", 20);
73      }
74
75      while (len = read(fd1, buffer, sizeof(buffer)))
76      {
77          int i;
78          for (i = 0; i < len; i++)
79          {
80              if (buffer[i] == '\n')
81              {
82                  ++lines;
83              }
84          }
85      }
86
87      close(fd1);
88      return lines;
89  }
```

Alínea e)

Função **apaga** ficheiro- Para apagar um ficheiro, tivemos de procurar uma system call que efetuasse essa ação. Encontramos a system call unlink, que apaga o ficheiro com o nome pedido.



```
91  /// Função utilizada para apagar um ficheiro
92  void apagaFicheiro(char fileName[50])
93  {
94      int fd;
95      fd = open(fileName, O_RDONLY);
96
97      if (fd == -1)
98      {
99          write(STDERR_FILENO, "Ficheiro não existe", 20);
100     }
101
102     unlink(fileName);
103
104     close(fd);
105 }
106
```

Alínea f)

Função **informa** ficheiro- Nesta função tínhamos como objetivo informar o utilizador do tipo de ficheiro que estava a abrir e quais as suas características.

Para isso utilizamos a system call stat. Esta função lê o ficheiro pretendido, e guarda a sua informação na variável file do tipo “struct stat” que criamos.

A partir desta variável foi possível listar todas as informações pedidas no enunciado.

Fizemos também uso das Macros “S_ISDIR”, “S_ISREG”, “S_ISLINK”, para verificar o tipo do ficheiro.

```
133  /// Função utilizada para obter a informação sobre um ficheiro
134  void informaFicheiro(char fileName[50])
135  {
136      struct stat file;
137      int fd;
138
139      stat(fileName, &file);
140
141      printf("st_mode = %d\n", file.st_mode);
142
143      if (fd = open(fileName, O_RDONLY) == -1)
144      {
145          write(STDERR_FILENO, "Ficheiro não existe", 20);
146      }
147      else
148      {
149          if (S_ISDIR(file.st_mode))
150          {
151              printf("\nDiretoria\n");
152          }
153          else if (S_ISREG(file.st_mode))
154          {
155              printf("\nFicheiro\n");
156          }
157          else if (S_ISLNK(file.st_mode))
158          {
159              printf("\nLink\n");
160          }
161
162          printf("\nI-node: %lo\n", file.st_ino);
163
164          printf("Utilizador dono: %o\n", file.st_uid);
165
166          printf("Data de criaçao: %s\n", ctime(&file.st_ctime));
167
168          printf("Data de leitura: %s\n", ctime(&file.st_atime));
169
170          printf("Data de modificaçao: %s\n", ctime(&file.st_mtime));
171      }
172  }
173
```

Alínea g)

Função **lista** diretoria: Para esta função fizemos uso de uma variável do tipo DIR, e uma do tipo dirent. Com a variável do tipo dirent foi possível percorrer cada elemento da diretoria, e imprimi-lo. Fizemos uso das macros usadas na alínea anterior, para verificar se era um ficheiro simples ou diretoria. Para abrir e fechar a diretoria usamos as funções opendir() e closedir().

```
171  /// Função utilizada para obter todos os ficheiros de uma determinada diretoria
172  void listaDiretoria(char path[100])
173  {
174      DIR* dir;
175      struct dirent *d;
176      struct stat file;
177
178
179      if(path[0]=='d' && path[1]=='\0')
180      {
181          dir = opendir(".");
182      }
183      else
184      {
185          dir = opendir(path);
186      }
187
188      if(dir == NULL)
189      {
190          write(STDERR_FILENO, "Diretoria não existe", 21);
191      }
192
193      while ((d=readdir(dir))!=NULL)
194      {
195          stat(d->d_name, &file);
196
197          if (S_ISDIR(file.st_mode))
198          {
199              printf("Diretoria\n");
200          }
201          else if (S_ISREG(file.st_mode))
202          {
203              printf("Ficheiro\n");
204          }
205          printf("->%s\n", d->d_name);
206      }
207
208      closedir(dir);
209  }
210
```

Menu

Menu: Para a melhor organização do código criamos um menu, usando o “switch case”. A variável “num” é avaliada e, em seguida, o “switch” compara o resultado do “num” com o valor de cada constante que segue cada um dos “case” e decide o rumo da ação.

```
219     write(STDOUT_FILENO, "\n1 - Mostra Ficheiro!\n", 22);
220     write(STDOUT_FILENO, "2 - Copia Ficheiro!\n", 21);
221     write(STDOUT_FILENO, "3 - Acrescentar ficheiro!\n", 27);
222     write(STDOUT_FILENO, "4 - Contar linhas!\n", 20);
223     write(STDOUT_FILENO, "5 - Apagar ficheiro!\n", 22);
224     write(STDOUT_FILENO, "6 - Informação ficheiro!\n", 27);
225     write(STDOUT_FILENO, "7 - Lista diretoria! Digite d para a diretoria atual.\n", 54);
226     write(STDOUT_FILENO, "Escolha: ", 10);
227     scanf("%d", &num);
228     switch (num)
229     {
230     case 1:
231         write(STDOUT_FILENO, "Insira o nome do ficheiro: ", 28);
232         scanf(" %s", fileName);
233         mostraFicheiro(fileName);
234         main();
235         break;
236     case 2:
237         write(STDOUT_FILENO, "Insira o nome do ficheiro que quer copiar: ", 44);
238         scanf(" %s", fileName);
239         copiarFicheiro(fileName);
240         main();
241         break;
242     case 3:
243         write(STDOUT_FILENO, "Insira o nome do ficheiro original: ", 37);
244         scanf(" %s", fileName);
245         write(STDOUT_FILENO, "Insira o nome do ficheiro para o qual quer copiar: ", 52);
246         scanf(" %s", fileNameCopy);
247         acrescentaFicheiro(fileName, fileNameCopy);
248         main();
249         break;
250
251     case 4:
252         write(STDOUT_FILENO, "Insira o nome do ficheiro: ", 28);
253         scanf(" %s", fileName);
254         printf("Linhas: %d\n", contaLinhas(fileName));
255         main();
256         break;
257
258     case 5:
259         write(STDOUT_FILENO, "Insira o nome do ficheiro que deseja apagar: ", 46);
260         scanf(" %s", fileName);
```

Parte 2

Introdução

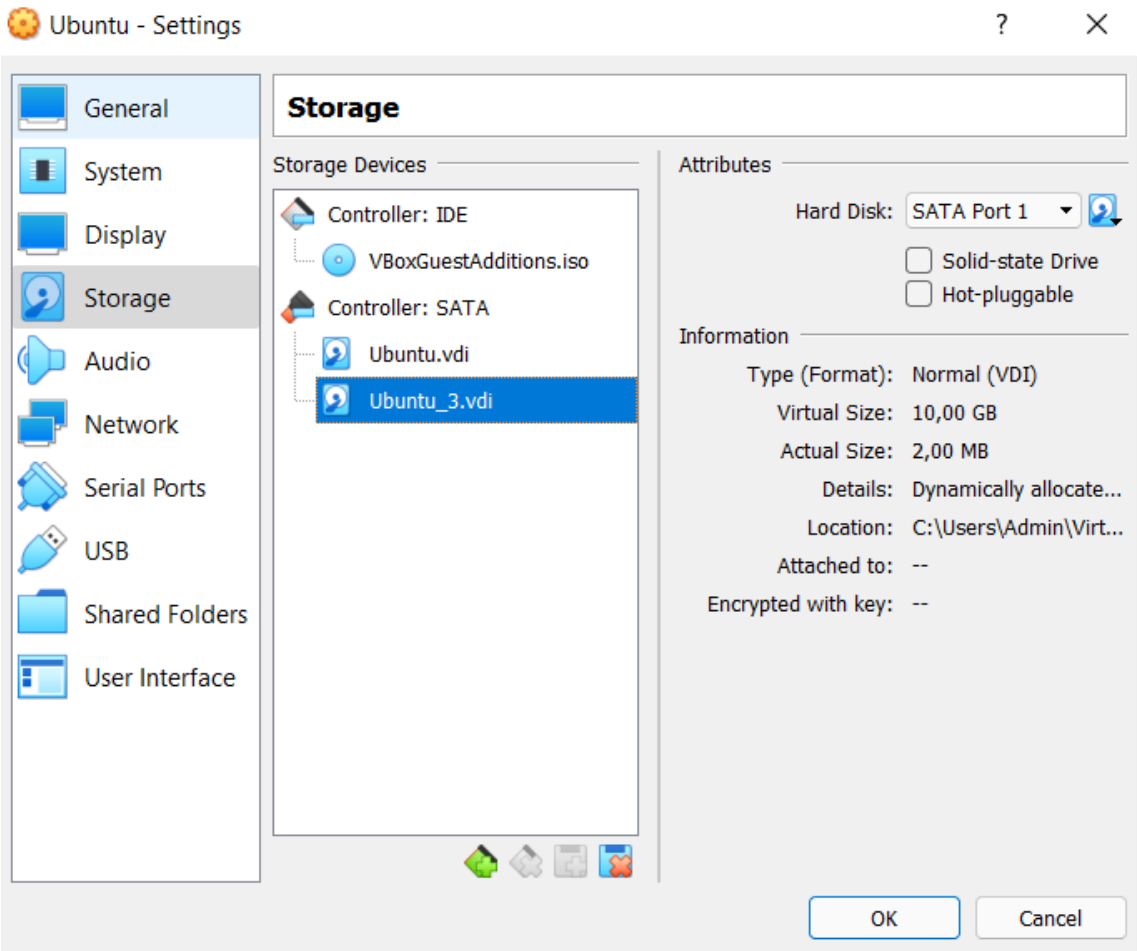
Para a parte 2 do trabalho prático, foi-nos proposta a implementação de alguns comandos para manipulação de ficheiros. O seguinte relatório irá documentar como foi efetuada a implementação, incluindo todas as etapas e dificuldades enfrentadas.

Resolução

Alínea a)

Nesta alínea tínhamos como objetivo criar na máquina virtual um disco novo com 10GB, e de seguida criar uma partição. As seguintes imagens mostram os comandos que e passos efetuados para cumprir esse objetivo:

Aqui criamos o disco necessário com 10GB, como foi pedido.



De seguida criamos uma partição dentro do disco.

```
joaofurtado8@joaofurtado8:~$ sudo fdisk -l /dev/sdb
[sudo] password for joaofurtado8:
Disk /dev/sdb: 10 GiB, 10737418240 bytes, 20971520 sectors
Disk model: VBOX HARDDISK
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
```

```
joaofurtado8@joaofurtado8:~$ sudo fdisk /dev/sdb

Welcome to fdisk (util-linux 2.36.1).
Changes will remain in memory only, until you decide to write them.
Be careful before using the write command.

Device does not contain a recognized partition table.
Created a new DOS disklabel with disk identifier 0x5dd0141d.

Command (m for help): n
Partition type
   p   primary (0 primary, 0 extended, 4 free)
   e   extended (container for logical partitions)
Select (default p): p
Partition number (1-4, default 1): 1
First sector (2048-20971519, default 2048):
Last sector, +/-sectors or +/-size{K,M,G,T,P} (2048-20971519, default 20971519):

Created a new partition 1 of type 'Linux' and of size 10 GiB.

Command (m for help): w
The partition table has been altered.
Calling ioctl() to re-read partition table.
Syncing disks.
```

Alínea b)

Nesta alínea tínhamos como objetivo criar um volume que ocupasse o espaço todo da partição, e lá adicionar 2 volumes lógicos cada um com 5GB.

Primeiro criamos o volume:

```
joaofurtado8@joaofurtado8:~$ sudo pvcreate /dev/sdb1
  Physical volume "/dev/sdb1" successfully created.
```

E de seguida um grupo de volumes dentro deste:

```
joaofurtado8@joaofurtado8:~$ sudo vgcreate vgsosd /dev/sdb1
  Volume group "vgsosd" successfully created
```

Para verificar que a criação ocorreu sem problemas, corremos os seguintes comandos:

```
joaofurtado8@joaofurtado8:~$ sudo pvs
  PV         VG     Fmt  Attr PSize   PFree
  /dev/sdb1  vgsosd lvm2 a--  <10,00g <10,00g
joaofurtado8@joaofurtado8:~$ sudo vgs
  VG     #PV #LV #SN Attr   VSize   VFree
  vgsosd   1   0   0 wz--n- <10,00g <10,00g
```

De seguida criamos os 2 volumes lógicos.

O primeiro volume a ser criado, foi criado com sucesso:

```
joaofurtado8@joaofurtado8:~$ sudo lvcreate -L 5G -n lvsosd vgsosd
  Logical volume "lvsosd" created.
```

Já o segundo, deu-nos a seguinte mensagem de erro: “volume group vgsosd has insufficient free space (1279 extents) 1280 required”. Então efetuamos o seguinte comando que criava o volume lógico com o espaço que estava livre para este:

```
joaofurtado8@joaofurtado8:~$ sudo lvcreate -l 100%FREE -n lvsosd2 vgsosd
  Logical volume "lvsosd2" created.
```

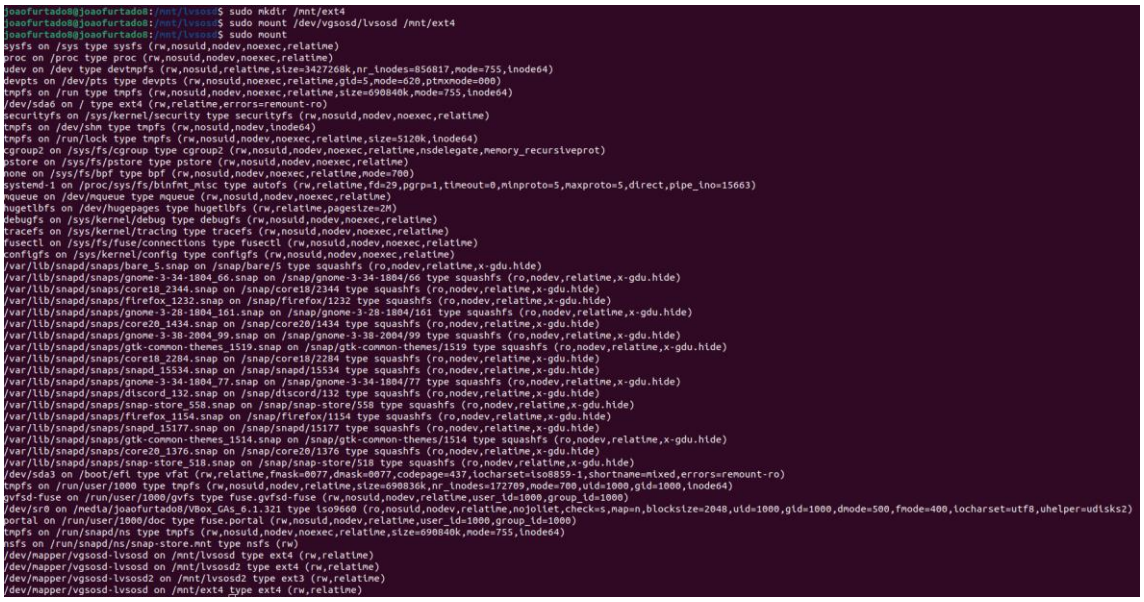
Alínea c)

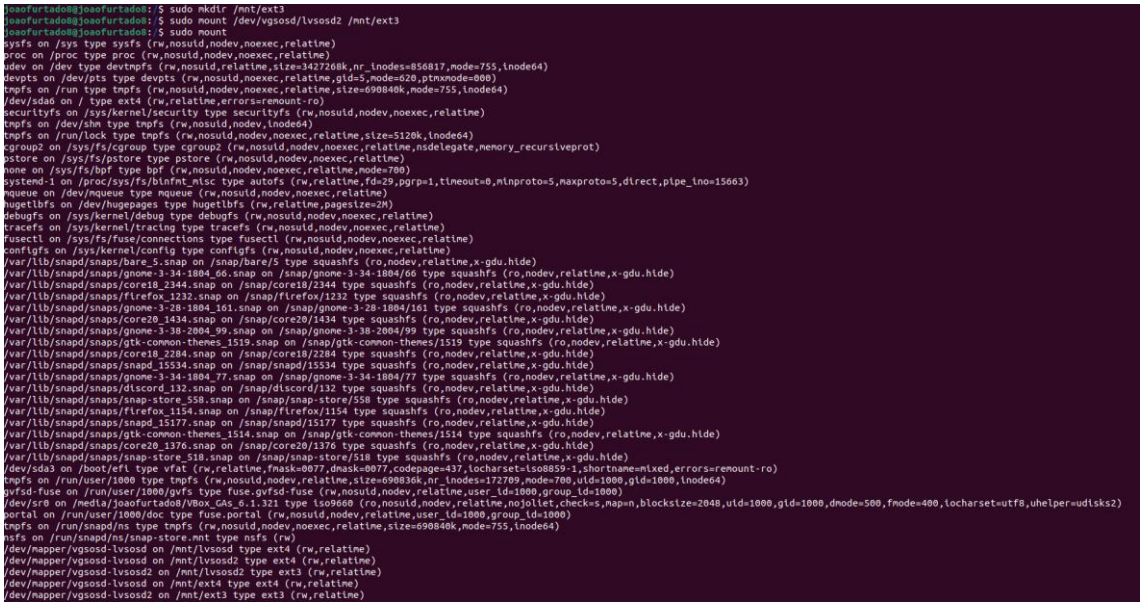
Nesta alínea procedemos à criação de um sistema de ficheiros ext4 no primeiro volume lógico, e um sistema de ficheiros ext3 no segundo.

```
joaofurtado8@joaofurtado8:~$ sudo mkfs.ext4 /dev/vgsosd/lvsosd
mke2fs 1.46.3 (27-Jul-2021)
Creating filesystem with 1310720 4k blocks and 327680 inodes
Filesystem UUID: 601239d5-c800-4c7b-9947-10c2e1690909
Superblock backups stored on blocks:
        32768, 98304, 163840, 229376, 294912, 819200, 884736

Allocating group tables: done
Writing inode tables: done
Creating journal (16384 blocks): done
Writing superblocks and filesystem accounting information: done

joaofurtado8@joaofurtado8:~$ sudo mkfs.ext3 /dev/vgsosd/lvsosd2
mke2fs 1.46.3 (27-Jul-2021)
Creating filesystem with 1309696 4k blocks and 327680 inodes
Filesystem UUID: 10f38d98-540e-4885-92ca-f871dd0a6714
Superblock backups stored on blocks:
        32768, 98304, 163840, 229376, 294912, 819200, 884736

Allocating group tables: done
Writing inode tables: done
Creating journal (16384 blocks): done
Writing superblocks and filesystem accounting information: done
```

Alínea d)

Nesta alínea, montámos os sistemas de ficheiros criados anteriormente nas diretorias /mnt/ext4 e /mnt/ext3.





Alínea e)

Depois de montar os sistemas de ficheiros, foi-nos pedido para um ficheiro na diretoria /mnt/ext4 com os números de aluno dos constituintes do grupo:

```
ruben@ruben-VirtualBox:/mnt/ext4$ sudo touch 21118-21125-21105-21135-2
1147-21143.txt
```

Com o ficheiro criado, tínhamos então de alterar as permissões do dono para escrita e leitura, retirar as permissões ao grupo, e dar permissão de leitura aos outros:

```
ruben@ruben-VirtualBox:/mnt/ext4$ sudo chmod u+rw 21118-21125-21105-21
135-21147-21143.txt
```

```
ruben@ruben-VirtualBox:/mnt/ext4$ ls -la 21118-21125-21105-21135-21147
-21143.txt
-rw-r--r-- 1 root root 0 mai  1 12:43 21118-21125-21105-21135-21147-21
143.txt
ruben@ruben-VirtualBox:/mnt/ext4$ sudo chmod g-r 21118-21125-21105-211
35-21147-21143.txt
ruben@ruben-VirtualBox:/mnt/ext4$ ls -la 21118-21125-21105-21135-21147
-21143.txt
-rw----r-- 1 root root 0 mai  1 12:43 21118-21125-21105-21135-21147-21
143.txt
```

Alínea f)

Na alínea final da Parte 2, verificamos quais as permissões efetivas que o ficheiro /etc/shadow tem:

```
joaofurtado8@joaofurtado8:/etc$ ls -la shadow
-rw-r----- 1 root shadow 1519 mar 17 16:22 shadow
```

Como podemos verificar, ninguém pode executar o seguinte ficheiro. Tanto o dono como o grupo podem ler o seguinte ficheiro. Apenas o dono pode escrever neste ficheiro.

Conclusão

Concluindo, com a realização deste trabalho prático adquirimos novos conhecimentos em relação às system calls e comandos de manipulação de ficheiros de linux assim como a criação de partições do disco. Foi um trabalho que demostrou ser bastante desafiante ao longo da sua execução, mas também uma boa forma de consolidar toda a matéria lecionada até ao momento.